Indécidabilité de la satisfiabilité d'une requête

Leçons : 914 ; 932

Références : ABITEBOUL–HULL–VIANU, *Foundations of Database* (p.123) [?] et LESESVRE–MONTAGNON–LE BARBENCHON–PIERRON, *131 développements pour l'oral* (p. 817) [?]

Notations :

- pour $u_{i_1} \dots u_{i_n}$, on note $u^{(k)}$ pour $u_{i_1} \dots u_{i_k}$ avec $k \in [\![1, n]\!]$
- pour $v_{i_1} \dots v_{i_n}$, on note $v^{(k)}$ pour $v_{i_1} \dots v_{i_k}$ avec $k \in [\![1, n]\!]$

Introduction :

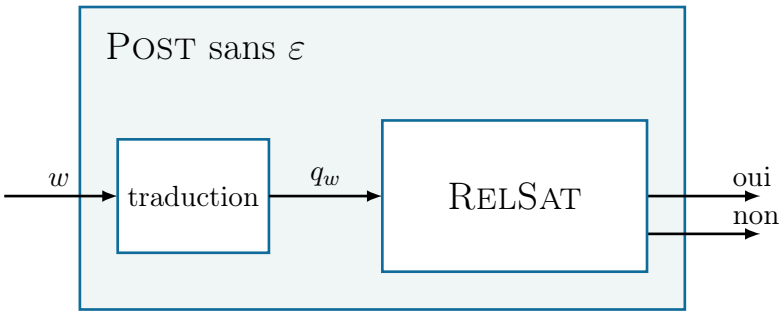
On veut montrer que le problème qui est de savoir si une requête du calcul relationnel est satisfiable est un problème indécidable. On va pour le prouver montrer une réduction du problème Post à RelSat.

Théorème 1. Le problème

RelSat { entrée : une requête q du calcul relationnel
sortie : oui si il existe une base de donnée $\mathcal{I}$ telle que $q(\mathcal{I}) \neq \emptyset$, non sinon

est indécidable.

Démonstration. On va réduire le problème Post sans ε [1] à RelSat.



Soit w une instance de Post, c'est à dire un nombre fini de tuiles $\boxed{\begin{matrix} u_1 \\ v_1 \end{matrix}}, \dots, \boxed{\begin{matrix} u_N \\ v_N \end{matrix}}$ sur l'alphabet $\Sigma = \{a, b\}$.

On prend le schéma de base de données suivant

$$\mathcal{R} = \{\text{Enc}(pos1, pos2, mot, u, v), \text{Sync}(debut\_u, debut\_v)\}$$

Je note Enc pour Encoder et Sync pour Synchroniser

Les colonnes *pos*1 et *pos*2 servent à mettre un ordre sur les tuples. Pour une instance positive de Post, on écrira le mot qui figure en haut et en bas d'une concaténation de tuiles valide dans la colonne *mot*. La colonne u (respectivement v) sert à donner l'indice de la tuile du haut (resp. du bas) à laquelle correspond la lettre qui est dans la colonne *mot*. (voir les exemples de remplissage du tableau plus bas)

1. Post sans ε { entrée : des tuiles du problème de Post sans tuile contenant des ε
sortie : oui si on peut trouver une suite de tuiles ayant le même mot en haut et en bas, non sinon

(pour la réduction du problème de Post original voir l'appendice ici).

On pose la requête suivante

$$q_w = \{\langle\rangle, \varphi_w\}$$

où φ_w est la conjonction des 5 formules suivantes :

— φ_{Enc} qui traduit le fait que
- les éléments de $pos1$ sont disjoints
- les colonnes $pos1, pos2$ forment un cycle de longueur > 1
- $pos1$ contient le symbole \$

Cette formule sert à forcer les tuples de la relation Enc à être dans un certain ordre [2] et donc de retrouver cet ordre même si une relation est un ensemble de tuples (donc sans ordre a priori). C'est pour ça que l'on a besoin de deux colonnes. La première colonne est une clé pour retrouver le début de chaque tuile (d'où la nécessité d'avoir des éléments distincts). Le symbole \$ est un symbole spécial pour marquer le début.

— φ_{Sync} qui traduit le fait que les éléments de $debut\_u$ sont disjoints et les éléments de $debut\_v$ sont disjoints

On a les indices de début de chaque tuile, donc il faut que les éléments soient distincts.

— φ_u qui traduit le fait que pour tout $x \in debut\_u$, il existe $i \in [\![1, N]\!]$ avec $u_i = z_1 \dots z_k$ et $x_1, \dots, x_{k+1}$ avec $x = x_1$ tels que

$$\langle x_1, x_2, z_1, i, * \rangle \in \text{Enc}$$

$$\langle x_2, x_3, z_2, i, * \rangle \in \text{Enc}$$

$$\vdots$$

$$\langle x_k, x_{k+1}, z_k, i, * \rangle \in \text{Enc}$$

avec $x_1, x_{k+1} \in debut\_u$ et $x_2, \dots, x_k \notin debut\_u$
(où $*$ peut être choisi indifféremment)

Cette formule permet de dire que pour chaque indice de début de tuile dans Sync, on a une tuile $\boxed{\begin{array}{|c|}\hline u_i \\ \hline v_i \\ \hline \end{array}}$ et que (dans l'ordre imposé par la formule φ_{Enc}) on a bien une succession de tuples qui correspond au mot u_i de la tuile $\boxed{\begin{array}{|c|}\hline u_i \\ \hline v_i \\ \hline \end{array}}$ puis que les lettres du mot correspondent bien à u_i entre les indices $x = x_1$ et x_k. De plus, on oblige à ne pas avoir de début de tuile pendant les indices intermédiaires, et que l'indice suivant (x_{k+1}) soit un début de tuile.

— φ_v similaire à φ_u
φ_v qui traduit le fait que pour tout $x \in debut\_v$, il existe $i \in [\![1, N]\!]$ avec $v_i = z_1 \dots z_k$ et $x_1, \dots, x_{k+1}$ avec $x = x_1$ tels que

$$\langle x_1, x_2, z_1, *, i \rangle \in \text{Enc}$$

$$\langle x_2, x_3, z_2, *, i \rangle \in \text{Enc}$$

$$\vdots$$

$$\langle x_k, x_{k+1}, z_k, *, i \rangle \in \text{Enc}$$

avec $x_1, x_{k+1} \in debut\_v$ et $x_2, \dots, x_k \notin debut\_v$
(où $*$ peut être choisi indifféremment)

2. l'ordre que l'on donne dans les tables que l'on dessine dans le développement, c'est en fait l'ordre dans lequel on lit le mot qui est en haut et en bas d'une concaténation valide de tuiles du problème Post

Cette formule permet de dire que pour chaque indice de début de tuile dans Sync, on a une tuile $\boxed{\begin{array}{|c|}\hline u_i \\ \hline v_i \\ \hline \end{array}}$ et que (dans l'ordre imposé par la formule φ_{Enc}) on a bien une succession de tuples qui correspond au mot v_i de la tuile $\boxed{\begin{array}{|c|}\hline u_i \\ \hline v_i \\ \hline \end{array}}$ puis que les lettres du mot correspondent bien à v_i entre les indices $x = x_1$ et x_k. De plus, on oblige à ne pas avoir de début de tuile pendant les indices intermédiaires, et que l'indice suivant (x_{k+1}) soit un début de tuile.

— $\varphi_{\text{Sync\_}u\_v}$ définie par

$$\begin{aligned}
&\textsc{Sync}(\$, \$) \\
&\wedge \forall x \forall y \ \Big(\textsc{Sync}(x,y) \to \bigvee_{i=1}^{N} \exists t_1 t_2 t_3 s_1 s_2 s_3 \textsc{Enc}(x, t_1, t_2, i, t_3) \wedge \textsc{Enc}(y, s_1, s_2 s_3, i) \Big) \\
&\wedge \forall x \forall y \ \Big(\textsc{Sync}(x,y) \ \wedge \ \neg(x = \max(\mathit{debut_}u)) \ \wedge \ \neg(y = \max(\mathit{debut_}v)) \to \textsc{Sync}(x', y') \Big)
\end{aligned}$$

avec $x' = \min\limits_{a \in \mathit{debut\_}u} a > x$ et $y' = \min\limits_{a \in \mathit{debut\_}v} a > y$

Cette formule force à commencer par une première tuile (par le fait que $\textsc{Sync}(\$, \$)$), puis à synchroniser le reste des tuiles, c'est-à-dire que après la tuile $\boxed{\begin{array}{|c|}\hline u_{i_1} \\ \hline v_{i_1} \\ \hline \end{array}}$, on accède à la tuile suivante (3ème sous-formule), et que c'est une tuile licite (car il faut que ce soit une tuile $\boxed{\begin{array}{|c|}\hline u_i \\ \hline v_i \\ \hline \end{array}}$ où l'indice du haut est le même que l'indice du bas et non une tuile $\boxed{\begin{array}{|c|}\hline u_3 \\ \hline v_4 \\ \hline \end{array}}$ qui ne serait pas licite) (par la 2ème sous-formule).

Il faut montrer que

$$\boxed{q_w \text{ est satisfiable SSI } w \text{ est une instance positive de } \textsc{Post}}$$

$\boxed{\Longrightarrow}$ Soit une instance $\mathcal{I}$ telle que $\mathcal{I} \vDash q_w$.
On note $r = |\textsc{Sync}|$ (ce sera le nombre de tuiles)
On veut montrer par récurrence sur $k \in [\![1, r]\!]$ que

$$HR_k : \begin{cases} u^{(k)} = mot[1 \dots |u^{(k)}|] \\ v^{(k)} = mot[1 \dots |v^{(k)}|] \end{cases}$$

On prouve cela grâce aux formules de φ_w qui est satisfiable [3].

Ainsi, on peut conclure que $u^{(r)} = v^{(r)}$ et donc que w est une instance positive de Post.

$\boxed{\Longleftarrow}$ w est une instance positive de Post, donc il existe $i_1, \dots, i_r$ tels que

$$m = u_{i_1} \dots u_{i_r} = v_{i_1} \dots v_{i_r}$$

On construit alors la table Enc et Sync comme ci-dessous

3. les arguments sont en fait les explications en gris sous chaque formule

ENC

$pos1$	$pos2$	mot	u	v
\$	1	m_1	i_1	i_1
1	2	m_2	i_1	i_1
⋮	⋮	⋮	i_1	i_1
$\lvert v^{(1)}\rvert$	$\lvert v^{(1)}\rvert + 1$	$m_{\lvert v^{(1)}\rvert}$	i_1	i_2
⋮	⋮	⋮	i_1	⋮
⋮	⋮	⋮	i_1	⋮
$\lvert u^{(1)}\rvert$	$\lvert u^{(1)}\rvert + 1$	$m_{\lvert u^{(1)}\rvert}$	i_2	⋮
⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮
$\lvert m\rvert - 2$	$\lvert m\rvert - 1$	⋮	⋮	⋮
$\lvert m\rvert - 1$	\$	$m_{\lvert m\rvert}$	i_r	i_r

SYNC

$debut\_u$	$debut\_v$
\$	\$
$\lvert u^{(1)}\rvert$	$\lvert v^{(1)}\rvert$
$\lvert u^{(2)}\rvert$	$\lvert v^{(2)}\rvert$
⋮	⋮
⋮	⋮
$\lvert u^{(r-1)}\rvert$	$\lvert v^{(r-1)}\rvert$

Cette instance de base de données $\mathcal{I}$ (sur le schéma de base de données $\mathcal{R}$) satisfait la requête q_w [4]. Il suffit de vérifier qu'elle vérifie chaque formule que l'on a énoncé.

□

Remarques :

Il existe d'autres problèmes indécidables sur les bases de données comme :

- *RelEqu* :
  - entrée : deux requêtes q et q' du calcul relationnel
  - sortie : oui si q et q' sont sémantiquement équivalentes, non sinon
- *RelSure* :
  - entrée : une requête q du calcul relationnel
  - sortie : oui si q est sûre, non sinon

Pour montrer l'indécidabilité de RELEQU, il suffit de réduire co-RELSAT à RELEQU. En effet, une requête est insatisfiable si et seulement si elle est équivalente à la requête vide.

Astuces de l'agrégatif :

- Ce développement est très long, je ne pense pas qu'il soit faisable d'écrire toutes les formules et de prouver la réduction tout en faisant comprendre au jury ce que l'on est en train de faire. J'ai fait le choix d'essayer de faire comprendre au jury la réduction en donnant le sens des formules.
- J'écris la base de données à droite du tableau dès que j'introduis ENC et SYNC en expliquant comment les remplir, pour que le sens des formules soit compréhensible quand je les énonce ensuite. Je dis que le but des formules est que la seule façon de remplir la base de données soit celle que j'ai présenté.
- Lors du développement je dis juste POST, en effet ce n'est qu'une variante de POST.
- Je mets les formules ci-dessous, il peut être intéressant de donner une formule d'unicité sur la formule φ_{ENC} pour convaincre le jury que l'on maitrise aussi l'écriture rigoureuse des formules.
- On a prouvé ici que ce problème n'est pas dans R, mais il est quand même dans RE. En effet, on peut dénombrer les bases de données qui sont des ensembles finis de tuples sur un domaine dénombrable.

4. i.e. $q_w(\mathcal{I}) \neq \emptyset$

— φ_{Enc}

$$\begin{aligned}
&\exists t\Big(\psi(\$,t)\wedge\neg(t=\$)\Big)\wedge\exists t\Big(\psi(t,\$)\wedge\neg(t=\$)\Big) && \text{existence \$}\\
&\wedge\ \forall x\forall y\exists z\Big(\psi(x,y)\to\psi(y,z)\Big) && \text{cycle}\\
&\wedge\ \forall x\forall y\exists z\Big(\psi(y,x)\to\psi(z,y)\Big) && \text{cycle}\\
&\wedge\ \forall x\forall y\forall z\Big((\psi(x,y)\wedge\psi(x,z))\to(y=z)\Big) && \text{unicité}\\
&\wedge\ \forall x\forall y\forall z\Big((\psi(y,x)\wedge\psi(z,x))\to(y=z)\Big) && \text{unicité}\\
&\wedge\ \forall x t_1t_2t_3t_4s_1s_2s_3s_4\Big((\text{Enc}(x,t_1,t_2,t_3,t_4)\\
&\qquad\wedge\text{Enc}(x,s_1,s_2,s_3,s_4))\to(t_1=s_1\\
&\qquad\qquad\wedge t_2=s_2\\
&\qquad\qquad\wedge t_3=s_3\\
&\qquad\qquad\wedge t_4=s_4)\Big) && \text{unicité } pos1
\end{aligned}$$

avec $\psi(x,y):=\exists z\exists t\exists s\text{Enc}(x,y,z,t,s)$

— φ_{Sync}

$$\begin{aligned}
&\forall x\forall y\forall z\Big((\text{Sync}(x,y)\wedge\text{Sync}(x,z))\to y=z\Big) && \text{unicité } debut_u\\
&\wedge\ \forall x\forall y\forall z\Big((\text{Sync}(x,y)\wedge\text{Sync}(z,y))\to x=z\Big) && \text{unicité } debut_v
\end{aligned}$$

— φ_u

$$\begin{aligned}
\forall x\Big(\exists y\text{Sync}(x,y)\to\Big(&\bigvee_{i=1}^{N}\big(\exists x_1x_2\dots x_{|u_i|}x_{|u_i|+1}j_1\dots j_{|u_i|}\\
&\big((\bigwedge_{k=1}^{|u_i|}\text{Enc}(x_k,x_{k+1},u_i[k],i,j_k))\\
&\wedge(x=x_1)\wedge(\exists z\text{Sync}(x_{|u_i|+1},z))\\
&\wedge\bigwedge_{k=2}^{|u_i|}\neg\exists y\text{Sync}(x_k,y)\big)\big)\Big)\Big)
\end{aligned}$$

— φ_v similaire

— $\varphi_{\text{Sync\_u\_v}}$ il faut coder $x<y$, pour cela il faut borner le nombre d'intermédiaire entre x et y, on n'a a priori pas de borne, mais on sait que l'indice de début de tuile suivante est forcément à moins de "taille maximum des mots sur les tuiles" de x, donc on peut faire un disjonction sur le nombre d'intermédiaires qui sera donc borné puisqu'on regarde toujours la différence entre x et y sur des tuiles qui se suivent. Pour faire l'égalité avec le maximum, il faut regarder si dans le tableau Enc, le dernier caractère de la tuile possède une $pos2$ égale à \$.

$$x<y:=\neg(x=y)\wedge\bigvee_{k=1}^{\text{borne}}\exists z_1z_2\dots z_k\big(\psi(x,z_1)\wedge\psi(z_1,z_2)\wedge\dots\wedge\psi(z_k,y)\big)$$

$$x=\max(debut\_u):=\bigvee_{k=0}^{|u_x|}\exists z_1\dots z_k\big(\psi(x,z_1)\wedge\psi(z_1,z_2)\wedge\dots\wedge\psi(z_k,\$)\big)$$

(Je ne suis pas sûr de ces dernières formules, il faudrait écrire vraiment rigoureusement tout cela pour être vraiment propre, mais je pense que cette idée marche [5]).

Je mets un exemple concret de tables pour $m = aabbbab = u_{i_1}u_{i_2}u_{i_3} = v_{i_1}v_{i_2}v_{i_3}$ avec les tuiles

$$i_1 \begin{array}{|c|}\hline aab \\ \hline a \\ \hline \end{array} ; i_2 \begin{array}{|c|}\hline b \\ \hline abbb \\ \hline \end{array} ; i_3 \begin{array}{|c|}\hline bab \\ \hline ab \\ \hline \end{array}$$

ENC

$pos1$	$pos2$	mot	u	v
\$	1	a	i_1	i_1
1	2	a	i_1	i_2
2	3	b	i_1	i_2
3	4	b	i_2	i_2
4	5	b	i_3	i_2
5	6	a	i_3	i_3
6	\$	b	i_3	i_3

SYNC

$debut\_u$	$debut\_v$
\$	\$
3	1
4	5

Questions possibles :

- Montrer que le problème de POST sans ε est indécidable.
- Écrire explicitement la récurrence de la question **c)** (i).

5. envoie moi un mail, si tu réussis à écrire ça proprement